Nonlinear Phenomena in Complex Systems, vol. **17**, no. 3 (2014), pp. 301 - 305

A Finite-State Model of Botnets' Desinfection and Removal

D. P. Zegzhda and T. V. Stepanova
Saint-Petersburg State Polytechnic University,
29 Politechnicheskaya Str, 195251 Saint-Petersburg, RUSSIA
(Received 28 March, 2014)

Existing multi-agent systems (either for implementing distributed security threats or for countering them in the Internet) either do not maintain agent graph connectivity or maintain it with network redundancy, which significantly increases the overheads. The paper analyzes a finite-state model of adaptive behavior in a multi-agent system. This model uses a d-regular agent graph and some methods to maintain network connectivity, which provides sustainable system performance in aggressive environment.

AMS Subject Classification: 05C80, 05C90
Keywords: multi-agent system, sustainable performance, adaptive behavior, random regular graph

1. Introduction

A multi-agent system is a system formed with multiple interacting intellectual agents. Generally, it consists of multiple organization units (agents), multiple tasks, environment, multiple agent relationships, multiple agent actions [1]. A multi-agent system can model an attacking party as well as a defending one. Both parties try to neutralize adversary's actions and maintain its own sustainability.

The paper represents the results of the analysis of sustainability maintaining in multi-agent systems countering distributed threats in the Internet. Performance sustainability is a complex aspect, and it describes the possibility of a system failure [2].

Not only functional but also topological system characteristics influence multi-agent system sustainability. There are two key parameters [3]:

1. Agent graph topology.
2. Protocol of agent interaction.

2. Topology analysis of multi-agent systems

Agent graph topology must fulfil several requirements, while creating a multi-agent security system:

1. the graph must have high connectivity parameters, while removing graph nodes and graph edges;
2. the network redundancy must be low.

Table 1 analyzes the topology of existing security systems and botnets from the point of view of these requirements [4].

Consequently, high connectivity in existing multi-agent systems (both attacking and defending ones) is provided with the increasing of network redundancy, which results in overheads increasing and system flexibility decreasing. There is an alternative approach, which suggests that fewer connections should be maintained, while using restore mechanisms for broken connections. If every node maintains a fixed number of d connections, the agent graph is a d-regular graph. While creating a system countering distributed threats in the Internet, the usage of a random d-regular agent graph is justified because the information available to the adversary diminishes. Though it is obvious that restore mechanisms are not able to restore the connection immediately, which results in unbalanced network distribution. Restoring of failing agents, which are rebuilding previous connections, will also contribute to it. Consequently, it is necessary to analyze almost

Table 1: Topology characteristics of the existing security systems.

Topology	Connectivity	Redundancy
Star	Low	Low
Star formed with other start	Medium	High
Scale-free graph	Medium	High
Smal world graph	Medium	High
Erdos and Renyi random graph	High	High

d-graph characteristics, where node degrees vary insignificantly, rather than d-regular graph characteristics.

3. Methods to provide sustainable performance

Some methods have been suggested to maintain network connectivity. They involve sending requests to get a new neighbor node address, if neighbors become fewer than some threshold number (regular graph vertex degree). Also the operating of neighbor nodes is periodically checked (node status check). While executing the checking algorithm, the first stage includes choosing an element from the list of neighbor addresses and sending there a status-checking message. The time of sending is fixed in a local database. Then the next neighbor address is chosen, and the process is repeated. The cycle continues until the whole list of neighbors is covered. If it is impossible to choose a neighbor address, it means that all the messages have been sent, and the algorithm competes itself.

The second algorithm is responsible for the network connectivity analysis. It tries to restore connectivity, if it is threatened. As a result, "dead" nodes are removed from the neighbor list. The algorithm tries to get the addresses of new nodes in the network. It chooses an address from the neighbor list. This address is then used as the key to address the local database in order to get the time of the request sending (node status checking request). If the time that has passed since the moment fixed in the database is more than some specified time, such node is removed from the neighbor list and added to the "dead" neighbor list. Otherwise, the algorithm moves to the next node. The information of the next node is checked. This approach allows removing from the neighbor list the nodes that have been disconnected for a long period. At the same time, because of the request algorithm those nodes that have some temporary connection issues are not removed. It happens because several messages are sent in order to duplicate failing ones.

"Dead" elements are removed from the neighbor list. After covering all the neighbors, the algorithm moves on to getting new neighbors. The number of the neighbors is calculated. If the neighbors are fewer than a certain number, the node needs getting new node addresses. If it is necessary to get new neighbors, the next neighbor is chosen, and a request is sent to establish new connection. If the request fails, the system tries to send it to the next neighbor. If it is impossible to send the request to any neighbor, the algorithm stops working. The next time it starts, the system tries to recover network connectivity. These algorithms are used to create a model of the adaptive behavior in a system shaped via a finite deterministic machine.

4. Finite-state machine of the system behavior

One or several states can be active in the active state configuration of an object's finite-state machine at any one time. If a state is active, there may start outgoing transition, which will result in an action and then activation of some other state(s). If there are more than one

Table 2: Agent states.

State	Description
Disconnected	A node is desconnected and it automatically starts connecting
Connecting	A node is connecting. The code of every node includes a list if adresses to connect
Getting new connections	A node is gettiong the adresses of new nodes
Enough connecting	A node has enough neighbor nodes
Ready	A node is ready to execute tasks
Reserved	A node is reserved for computing
Computing	A node is computing
Idle	A node is not waiting or processing reservation and coputing results
Reserving	Node computing powers are reserved
Waiting	The system is processing reserving or computing results
Processing	The system is processing reserving or computing results
Dispatching	Tasks are distributed among neighbor nodes

Table 3: Finite-state machine alphabet.

Event	Description
One of the following messages has been received or sent:	
Compute	task execution request
Compute result	task execution result
Reserve	Node reservation request
Reserve denied	Reservation rejection
Info	informative message
Change server_status	Change of master node status
Keep alive	Neighbor node operating check
Get new server	New connection request
Sent new server	sending the adress of a new neighbor node
Disconnect	Disconnection
One of the following timers has started working:	
Keep alive limit	Neighbor node operation check
Keep alive wait limit	Waiting for an operation confirmation
Reserved.timeout	a reserved node waiting for the task
idle.timeout	being idle
reserving.list[i].wait limit	waiting for the response for a reservation request
The value of one the following counters has become more or less than some threshold value :	
neighbors.size	neighbor node list capacity
task.size	task size
status= reserver, ready, off	node status
server status=server, node	node master status
task[i].status=ready, waiting	task status
reserving.list[i].status=reserved, waiting	reservation status

active state, there is internal parallelism. Parallel states have some limitations, defined by a finite-state machine structure and its transitions. If a sequential composite state is active, only one disjoint substate can be active. If a parallel composite state is active, each of its parallel substates should be active.

The finite-state machine suggested in the

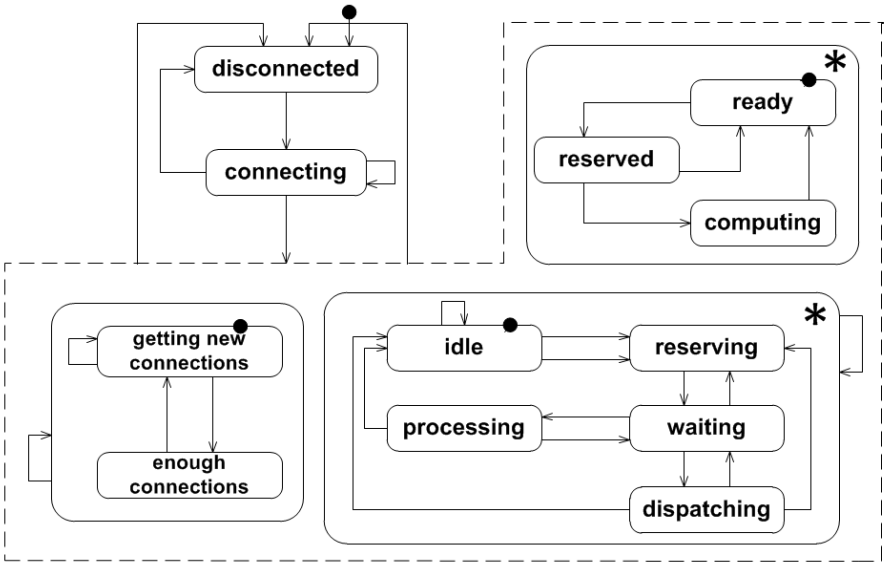

FIG. 1: Transition graph.

project involves parallel composite states. One of them is responsible for network connectivity; the other is responsible for computing. The agent can be in one of the following states (see Table 2).

The machine thus is defined by the tuple $M = (Q, \sum, \delta, q_0, F)$. Multiple machine states are $Q = \{$*disconnected, connecting, getting new connections, enough connections, ready, reserved, computing, idle, reserving, waiting, processing, dispatching*$\}$, the initial state is $q_0 =$ *disconnected*, multiple finite states F are empty, because, while transiting into the *disconnected* state, the system is trying to reconnect. An acceptable input alphabet $\sum$ includes possible transmitted messages, counter edge states, node statuses, tasks, and reservations (see Table 3).

The transition between the states is activated either by a received message or by the change of counter or timer state (e.g. counter that checks neighbor node statuses). Machine transition function $\delta : Q \times \Sigma \rightarrow P(Q)$ is described by a transition graph in Fig 1. The substates of composite states are united with a loop (e.g. ready, reserved, computing). Several parallel states are marked with dash-lines uniting parallel states. If a transition between states is marked with outgoing arrows, the transition takes place from any state. An ingoing arrow means returning to the state previous to the change. The states marked with an asterisk (*) are multiple composite ones. It means that a state is a set of parallel equal composite states, each of them describing the processing of one of the tasks given to a node. The active substate of the composite state is defined by the task state.

An agent graph vertex degree thus can be expressed as $\sigma(v) = d \pm \varepsilon$. This graph is almost a d-regular one, and we can make some suggestions in this case that are true for regular graphs. Particularly we can introduce a concept of extreme probability P_c. It must be such that, if $p < (1 - \delta)$, for every $\varepsilon > 0$ there is no graph component with minimal vertex number εn with probability $1 - o(1)$. This concept corresponds to extreme probability concept in the percolation theory. If $p > P_c(1 + \delta)$, there is a graph component containing minimum vertex ε for a $\varepsilon > 0$ with probability $1-o(1)$. For a random d-regular graph it is true that $P_c^{site} = P_c^{bond} = \frac{1}{d-1}$, where P_c^{site} is the extreme probability of graph vertex removing, and P_c^{bond} is the extreme probability of edge removing [5, 6].

5. Conclusion

The finite-state model of adaptive behavior in a multi-agent system countering distributed threats in the Internet has been suggested. It maintains d-regular agent graph characteristics using the methods that maintain system sustainability. It is an alternative approach compared with existing multi-agent system models that provide agent graph connectivity using network redundancy.

References

[1] V.B. Tarasov. *From multi-agent systems to intellectual organizations*. (Editorial URSS, Moscow., 2002). 352.

[2] D.P. Zegzhda, T.V. Stepanova. Effectiveness analysis of the security measures for botnet neutralizing and eliminating. Problems of Information Security, Computer Systems. **2**, 21-27 (2012).

[3] D.P. Zegzhda, T.V. Stepanova. Effectiveness analysis of the security measures against targeted botnet attacks. In: *The Proceedings of the XII International Information Security Research and Application Conference.* Part I. 2012.

[4] D. Dagon, G. Gu, C. Zou, J. Grizzard, S. Dwivedi, W. Lee, R. Lipton. *A Taxonomy of Botnets*. 2010.

[5] N. Fountoulakis. *Percolation on sparse random graphs with given degree sequence*. 2007.

[6] M. Molloy, B. Reed. *A critical point for random graphs with a given degree sequence*. 2000. http://www.math.mcmaster.ca/tom/Research Papers/MollReed95.pdf